A Robust Security Mechanism for Mobile Commerce Transactions

SUMMARY In 2006, Yeh and Tsai proposed a mobile commerce security mechanism. However, in 2008, Yum et al. pointed out that Yeh-Tsai security mechanism is not secure against malicious WAP gateways and then proposed a simple countermeasure against the attack is to use a cryptographic hash function instead of the addition operation. Nevertheless, this paper shows that both Yeh-Tsai’s and Yum et al.’s security mechanisms still do not provide perfect forward secrecy and are susceptible to an off-line guessing attack and Denning-Sacco attack. In addition, we propose a new security mechanism to overcome the weaknesses of the previous related security mechanisms.

key words: cryptography, security analysis, security protocol, mobile commerce, WAP, authentication

1. Introduction

Due to the mobile devices have become popular in recent years, mobile electronic transactions over the mobile platform is also growing fast [1]. In this environment, secure mobile electronic transactions between the mobile client and the mobile commerce server should be guaranteed for protecting mobile commerce transactions. Moreover, due to resource constraints of mobile computing platforms, lightweight security mechanisms are needed for protecting mobile commerce transactions [2].

In 2003, Lam et al. [3] proposed a lightweight security mechanism for protecting mobile transactions, which was designed to meet the security needs in face of the resource constraints. However, the lightweight security mechanism depends on the assumption that the mobile client should have the mobile commerce server’s public key in advance. To overcome the drawback and gain more efficiency, in 2006, Yeh and Tsai [4] proposed an enhanced mobile commerce security mechanism. The main idea of Yeh-Tsai’s security mechanism is to utilize the WAP gateway instead of the mobile client to verify the mobile commerce server’s public key. Through the security analysis, Yeh-Tsai’s claimed that their mechanism ensures that even a malicious WAP gateway cannot get the mobile client’s PIN by sending a faking public key. However, in 2008, Yum et al. [5] pointed out that Yeh-Tsai security mechanism is not secure against malicious WAP gateways by amplifying information leakage in addition operation. And also, they proposed a simple countermeasure against the attack is to use a cryptographic hash function instead of the addition operation.

Nevertheless, this paper shows that both Yeh-Tsai’s and Yum et al.’s security mechanisms still do not provide perfect forward secrecy [6], [7] and are susceptible to an offline guessing attack [8], [9] and Denning-Sacco attack [10]. Perfect forward secrecy means that if a long-term private key (e.g. user password or server private key) is compromised, this does not compromise any earlier session keys. A guessing attack involves an adversary (randomly or systematically) trying long-term private keys (e.g. user password or server secret key), one at a time, in the hope of finding the correct private key. Ensuring long-term private keys chosen from a sufficiently large space can reduce exhaustive searches. Most users, however, select passwords from a small subset of the full password space. Such weak passwords with low entropy are easily guessed by using the so-called dictionary attack. The Dennig-Sacco attack is where an attacker compromises an old session key and tries to find a long-term private key (e.g. user password or server private key) or other session keys. In addition, we propose a robust security mechanism for mobile commerce transactions to overcome the weaknesses of the previous related security mechanisms. As a result, the proposed security mechanism not only remedies the weaknesses shown in both security mechanisms, but also greatly improves the robustness of security mechanism through secure mutual authentication and session key agreement.

The remainder of this paper is organized as follows. In Sects. 2 and 3, we briefly review both Yeh-Tsai’s and Yum et al.’s security mechanisms and then describes its weaknesses. A robust security mechanism for mobile commerce transactions is proposed in Sect. 4 and the security discussions and the efficiency discussions are described in Sects. 5 and 6, respectively. Finally, we make some conclusions in Sect. 7.

2. Cryptanalysis of Yeh-Tsai’s Security Mechanism

This section briefly review the Yeh-Tsai’s security mechanism [4] and then shows that their security mechanism is not only susceptible to an off-line password guessing attack and Denning-Sacco attack, but also does not provide perfect forward secrecy [6]–[10]. The notations used throughout the paper can be summarized in Table 1. Figure 1 shows the
system architecture of security mechanism for secure mobile commerce transactions [3].

2.1 Review of Yeh-Tsai’s Security Mechanism

Figure 2 shows the Yeh-Tsai’s security mechanism and it performs as follows:

1. $S \rightarrow GW: R_a \oplus H(PIN), Cert$
   - S generates a random number $R_a$ and then XORed with $H(PIN)$ retrieved from its database. The result together with S’s certificate $Cert$ is sent to $GW$.

2. $GW \rightarrow C: R_a \oplus H(PIN), EKS$
   - After successfully verifying $Cert$ of $EKS$, $GW$ sends $S$’s public key $EKS$ together with the received $R_a \oplus H(PIN)$ to $C$.

3. $C \rightarrow GW \rightarrow S: EKS[(R_a + 1) \oplus PIN, R_b]$
   - $C$ gets $PIN$ from the user’s input and computes $H(PIN)$, which in turn is XORed with the received $R_a \oplus H(PIN)$ to get $R_a$. Then, $R_a$ is increased by one and then is XORed with $PIN$. The result and $R_b$, a random number generated by $C$, are encrypted with $EKS$ and sent to $S$ via the $GW$.

4. $S \rightarrow GW \rightarrow C: E_{sk}[SN, R_b]$
   - $S$ decrypts the received $EKS[(R_a + 1) \oplus PIN, R_b]$ by using its private key $DKS$ and then XORed the decrypted $(R_a + 1) \oplus PIN$ with $(R_a + 1)$ to get $PIN$. The hashed value $H(PIN)$ is then compared with the stored value $H(PIN)$ in the database for user authentication. If it holds, $S$ believes that the responding part is the real client; a serial number $SN$ and $R_b$ are encrypted with a session key $sk = R_a \oplus R_b$ and sent to $C$ via the $GW$ as a response. Otherwise, $S$ believes that the responding part is not the real client and the protocol is terminated.

5. Upon receiving the message $E_{sk}[SN, R_b]$ from $GW$, $C$ computes the shared secret key $sk = R_a \oplus R_b$ and then decrypts $E_{sk}[SN, R_b]$ to get $SN$ and $R_b$. $C$ checks $R_b$ is equal to its generated $R_b$. If so, $C$ believes that the responding part is the real server; otherwise, $C$ believes that the responding part is not the server and the proto-

![System architecture for mobile commerce transactions.](image1)

![Yeh-Tsai’s security mechanism.](image2)
col is terminated.

As a result, C and S can use the shared secret session key $sk = R_a \oplus R_b$ in private communication soon.

2.2 Off-Line Password Guessing Attack

This subsection shows that Yeh-Tsai’s security mechanism is vulnerable to off-line password guessing attacks[8], [9]. Let Eve be an active adversary who interposes the communication between C and GW. Then, Eve can easily obtain a legitimate communication parties’ password PIN by performing the following off-line password guessing attacks:

1. Eve → C: $X, EK_s^c$  
   When GW sends $R_a \oplus H(PIN)$ and $EK_s^c$ to C, Eve replaces $R_a \oplus H(PIN)$ with a random number $X$ and $EK_s^c$ with a fake public key $EK_s^f$. Finally, Eve sends $X$ and $EK_s^c$ to C.

2. $C \rightarrow Eve$: $EK_s^c[(X \oplus H(PIN)) + 1] \oplus PIN, R_b$  
   Upon receiving $X$ and $EK_s^c$, C will get PIN from the user’s input and compute $H(PIN)$, which in turn is XORed with the received $X$ to get $X \oplus H(PIN)$ Then, $X \oplus H(PIN)$ is increased by one and is XORed with PIN. The result $(X \oplus H(PIN)) + 1 \oplus PIN$ and a generated random number $R_b$ are encrypted with $EK_s^c$ and sent to Eve.

3. Eve decrypts $EK_s^c[(X \oplus H(PIN)) + 1] \oplus PIN, R_b$ by using the corresponding private key $DK_s^c$ of $EK_s^c$ to get $(X \oplus H(PIN)) + 1 \oplus PIN$ and $R_b$.  

4. Eve makes a guess at the secret password PIN* from dictionary $D$ to obtain the secret password PIN shared between C and S.

5. By using the decrypted value $(X \oplus H(PIN)) + 1 \oplus PIN$, Eve checks if $(X \oplus H(PIN)) + 1 \oplus PIN = (X \oplus H(PIN^*)) + 1 \oplus PIN^*$. If it holds, Eve has guessed the correct secret password PIN* = PIN.  

6. If it is not correct, Eve repeatedly performs the Steps (4) and (5) until $(X \oplus H(PIN)) + 1 \oplus PIN = (X \oplus H(PIN^*)) + 1 \oplus PIN^*$.  

The algorithm of an off-line password guessing attack is as follows:

Password Guessing Attack ($(X \oplus H(PIN)) + 1) \oplus PIN, X, D$

for $i := 0$ to $|D|$  

\[
\begin{align*}
PIN^* &\leftarrow D; \\
\text{if } (X \oplus H(PIN)) + 1 \oplus PIN^* = (X \oplus H(PIN^*)) + 1 \oplus PIN^* \\
\text{then return } PIN^*
\end{align*}
\]

2.3 Perfect Forward Secrecy Problem

Perfect forward secrecy [8] is a very important security requirement for evaluating a strong authentication protocol. An authentication protocol with perfect forward secrecy as-sures that even if one entity’s long-term key (e.g. user’s password or server’s secret key) is compromised, it will never reveal any old fresh session keys used before. For example, the well-known Diffie-Hellman key agreement scheme can provide perfect forward secrecy.

Yeh-Tsai’s security mechanism, however, does not provide it because once the secret password PIN of the client and the secret key $DK_s^c$ of the server are disclosed, all previous fresh session keys $sk = R_a \oplus R_b$ will also be opened and hence previous communication messages will be learned. In Yeh-Tsai’s security mechanism, suppose an attacker Eve obtains the secret password $H(PIN)$ and the secret private key $DK_s$ from the compromised server and intercepts transmitted values $(R_a \oplus H(PIN), EK_s^c[(R_a + 1) \oplus PIN, R_b])$ from an open network. It is easy to obtain the information since its are exposed over an open network. Then, Eve can compute $R_a \oplus H(PIN) \oplus H(PIN)$ by using the compromised $H(PIN)$ to get $R_a$ and decrypt $EK_s^c[(R_a + 1) \oplus PIN, R_b]$ by using the compromised $DK_s$ to get $R_b$. Finally, Eve can compute the shared session key $sk = R_a \oplus R_b$ by using $R_a$ and $R_b$. By using the compromised sk, Eve can get all previous communication messages. Obviously, Yeh-Tsai’s security mechanism does not provide perfect forward secrecy.

2.4 Denning-Sacco Attack

Denning-Sacco attack [10] is an offensive action where an attacker captures a session key from an eavesdropped session and uses the key either to gain the ability to impersonate the user directly or to mount a dictionary attack on the user’s password. Yeh-Tsai’s security mechanism is vulnerable to the Denning-Sacco attack based on a compromised session key $sk = R_a \oplus R_b$.

In the Yeh-Tsai’s security mechanism, suppose an attacker Eve obtains the session key $sk = R_a \oplus R_b$ from the compromised client or mobile commerce server and intercepts transmitted values $(R_a \oplus H(PIN), E_{sk}[SN, R_b])$ from an open network. It is easy to obtain this information since it is readily available over the open network. Then, Eve can decrypt $E_{sk}[SN, R_b]$ by using $sk$ to get $R_b$ and directly extract the hashed user’s secret password $H(PIN)$ by computing $R_a \oplus H(PIN) \oplus sk \oplus R_b$ as follows:

\[
R_a \oplus H(PIN) \oplus sk \oplus R_b = R_a \oplus H(PIN) \oplus R_a \oplus R_b \oplus R_b = H(PIN)
\]

(1)

Furthermore, if Eve wants to get real secret password PIN, he/she can obtain the PIN by performing the following off-line password guessing attack; Eve makes a guess at the secret password PIN* from dictionary D and then checks whether $H(PIN) = H(PIN^*)$. If it holds, Eve has guessed the correct secret password PIN* = PIN. If it is not correct, Eve repeatedly performs the verification process until $H(PIN) = H(PIN^*)$.

As a result, the compromise of the user’s secret password PIN or its hashed value $H(PIN)$ will enable the at-
tacker to impersonate the client C or the server S freely. Obviously, Yeh-Tsai’s security mechanism is insecure against a Denning-Sacco attack.

3. Cryptanalysis of Yum et al.’s Security Mechanism

This section briefly review the Yum et al.’s security mechanism and then shows that their security mechanism is not only susceptible to an off-line password guessing attack and Denning-Sacco attack, but also does not provide perfect forward secrecy.

3.1 Review of Yum et al.’s Security Mechanism

Figure 3 shows the Yum et al.’s security mechanism and it performs as follows:

1. \( S \rightarrow GW: R_a \oplus H(PIN), Cert \)
   This step is same as Yeh-Tsai’s security mechanism.
2. \( GW \rightarrow C: R_a \oplus H(PIN), EK_S \)
   This step is same as Yeh-Tsai’s security mechanism.
3. \( C \rightarrow GW \rightarrow S: EK_S[H(R_a), R_b] \)
   C gets PIN from the user’s input and computes \( H(PIN) \), which in turn is XORed with the received \( R_a \oplus H(PIN) \) to get \( R_a \). Then, \( H(R_a) \) and \( R_b \), a random number generated by C, are encrypted with \( EK_S \) and sent to S via the GW.
4. \( S \rightarrow GW \rightarrow C: E_{sk}[SN, R_b] \)
   S decrypts the received \( EK_S[H(R_a), R_b] \) by using its private key \( DK_S \) and then the hashed value \( H(R_a) \) is then compared with the compute value \( H(R_a) \) for user authentication. If it holds, S believes that the responding part is the real client; a serial number \( SN \) and \( R_b \) are encrypted with a session key \( sk = R_a \oplus R_b \) and sent to C via the GW as a response. Otherwise, S believes that the responding part is not the real client and the protocol is terminated.
5. Upon receiving the message \( E_{sk}[SN, R_b] \) from GW, C computes the shared session key \( sk = R_a \oplus R_b \) and then decrypts \( E_{sk}[SN, R_b] \) to get \( SN \) and \( R_b \). C checks \( R_b \) is equal to its generated \( R_b \). If so, C believes that the responding part is the real server; otherwise, C believes that the responding part is not the server and the protocol is terminated.

As a result, C and S can use the shared secret session key \( sk = R_a \oplus R_b \) in private communication soon.

3.2 Oﬄine Password Guessing Attack

This subsection shows that Yum et al.’s security mechanism is also vulnerable to oﬄine password guessing attacks. Let \( Eve \) be an active adversary who interposes the communication between C and GW. Then, \( Eve \) can easily obtain a legitimate communication parties’ password PIN by performing the following off-line password guessing attacks:

1. \( Eve \rightarrow C: X, EK_S^* \)
   When GW sends \( R_a \oplus H(PIN) \) and \( EK_S \) to C, \( Eve \) replaces \( R_a \oplus H(PIN) \) with a random number \( X \) and \( EK_S \) with a fake public key \( EK_S^* \). Finally, \( Eve \) sends \( X \) and \( EK_S^* \) to C.
2. \( C \rightarrow Eve: EK_S^*[X \oplus H(PIN), R_b] \)
   Upon receiving \( X \) and \( EK_S^* \), C will get PIN from the user’s input and compute \( H(PIN) \), which in turn is XORed with the received \( X \) to get \( X \oplus H(PIN) \). Then, the hash value of \( X \oplus H(PIN) \) and a generated random number \( R_b \) are encrypted with \( EK_S^* \) and sent to \( Eve \).
3. \( Eve \) decrypts \( EK_S^*[X \oplus H(PIN), R_b] \) by using the corresponding private key \( DK_S^* \) of \( EK_S^* \) to get \( X \oplus H(PIN) \)
and $R_b$.

4. Eve makes a guess at the secret password PIN from the dictionary $D$ to obtain the secret password PIN shared between $C$ and $S$.

5. By using the decrypted value $X \oplus H(PIN)$, Eve checks if $X \oplus H(PIN) = X \oplus H(PIN^*)$. If it holds, Eve has guessed the correct secret password PIN.

6. If it is not correct, Eve repeatedly performs the steps (4) and (5) until $X \oplus H(PIN) = X \oplus H(PIN^*)$.

The algorithm of an off-line password guessing attack is as follows:

**Password Guessing Attack** ($(X \oplus H(PIN)) + 1) \oplus PIN, X, D)$

\[
\begin{align*}
&\text{for } i := 0 \text{ to } |D| \\
&PIN^* \leftarrow D; \\
&\text{if } X \oplus H(PIN) = X \oplus H(PIN^*) \\
&\text{then return } PIN^*
\end{align*}
\]

3.3 Perfect Forward Secrecy Problem

Like the Yeh-Tsai’s security mechanism, Yum et al.’s security mechanism also does not provide the perfect forward secrecy because once the secret password PIN of the client and the secret key $DK_S$ of the server are disclosed, all previous fresh session keys $sk = R_a \oplus R_b$ will also be opened and hence previous communication messages will be learned.

In the Yum et al.’s security mechanism, suppose an attacker Eve obtains the secret password $H(PIN)$ and the secret private key $DK_S$ from the compromised server and intercepts transmitted values ($R_a \oplus H(PIN)$, $EK_S[H(R_a), R_b]$) from an open network. It is easy to obtain the information since its are exposed over an open network. Then, Eve can compute $R_a \oplus H(PIN) \oplus H(PIN)$ by using the compromised $H(PIN)$ to get $R_a$ and decrypt $EK_S[H(R_a), R_b]$ by using the compromised $DK_S$ to get $R_b$. Finally, Eve can compute the shared session key $sk = R_a \oplus R_b$ by using $R_a$ and $R_b$. By using the compromised $sk$, Eve can get all previous communication messages. Obviously, Yum et al.’s security mechanism does not provide perfect forward secrecy.

3.4 Denning-Sacco Attack

Since Yum et al.’s security mechanism performs basically the same process as Yeh-Tsai’s except the steps (3) and (4), Yum et al.’s security mechanism is also vulnerable to the Denning-Sacco attack based on a compromised session key $sk = R_a \oplus R_b$. The attack procedure is same as described in the above Sect. 2.4. So, we omit the detail description of the attack procedure in here. For further details, please refer to the above Denning-Sacco attack.

4. Proposed Security Mechanism

This section proposes a robust security mechanism that can not only withstand the off-line password guessing attack and Denning-Sacco attack, but also provide perfect forward secrecy. Figure 4 shows the proposed security mechanism and it performs as follows:

1. $S \rightarrow GW$: $aP \oplus H(PIN), Cert$

   $S$ randomly selects a number $a \in Z_n^*$, computes $aP$, and then XORed it with $H(PIN)$ retrieved from its database. The result together with $S$’s certificate $Cert$ is sent to $GW$.

![Fig. 4 Proposed security mechanism.](image)
2. **GW → C**: $aP \oplus H(PIN), EK_S$
   After successfully verifying Cert of $EK_S$, GW sends $S$’s public key $EK_S$ together with the received $aP \oplus H(PIN)$ to C.

3. **C → GW → S**: $EK_S[bP, H(sk)]$
   C gets PIN from the user’s input and computes $H(PIN)$. Then, $H(PIN)$ is XORed with the received $aP \oplus H(PIN)$ to get $aP$. C randomly selects a number $b \in Z_n$ and computes $bP$ and a session key $sk = abP$. Finally, $bP$ and $H(sk)$ are encrypted with $EK_S$ and sent to S via the GW.

4. **S → GW → C**: $E_{sk}[SN, bP]$
   S decrypts the received $EK_S[bP, H(sk)]$ by using its private key $DK_S$ and then computes a session key $sk = abP$ and its hash value $H(sk)$. The hashed value $H(sk)$ is then compared with the decrypted value $H(sk)$ for user authentication. If it holds, S believes that the responding part is the real client; a serial number $SN$ and $bP$ are encrypted with a session key $sk$ and sent to C via the GW as a response. Otherwise, S believes that the responding part is not the real client and the protocol is terminated.

5. Upon receiving the message $E_{sk}[SN, bP]$ from GW, C decrypts $E_{sk}[SN, bP]$ by using $sk$ to get $SN$ and $bP$. Then, C checks $bP$ is equal to its generated $bP$. If so, C believes that the responding part is the real server; otherwise, C believes that the responding part is not the server and the protocol is terminated.

   As a result, C and S can use the shared secret session key $sk = abP$ in private communication soon.

5. **Security Analysis**

This section analyzes the security of the proposed security mechanism. First, we define the security terms [7], [8] needed to conduct an analysis of the proposed security mechanism. They are as follows:

**Definition 1**: A weak secret key (user’s password PIN) is the value of low entropy $W(k)$, which can be guessed in polynomial time.

**Definition 2**: A strong secret key (server’s private secret key $DK_S$) is the value of high entropy $S(k)$, which cannot be guessed in polynomial time.

**Definition 3**: The Elliptic Curve Discrete Logarithm Problem (ECDLP) is as follows: given a public key point $V = aP$, it is hard to compute the secret key $a$.

**Definition 4**: The Elliptic Curve Diffie-Hellman Problem (ECDHP) is as follows: given point elements $aP$ and $bP$, it is hard to find $aPbP$.

**Definition 5**: A secure one-way hash function $y = H(x)$ is one where given $x$ to compute $y$ is easy and given $y$ to compute $x$ is hard.

Here, the following six security properties [6]–[10] must be considered for the proposed security mechanism: replay attacks, man-in-middle attacks, modification attacks, password guessing attacks, Denning-Sacco attacks, mutual authentication, and perfect forward secrecy. Regarding the above mentioned definitions, the followings are used to analyze the six security properties of the proposed security mechanism.

5.1 **Resistance to Replay Attacks**

The proposed security mechanism can resist replay attacks: A replay attack is an offensive action in which an attacker impersonates or deceives another legitimate participant through the reuse of information obtained in a protocol. When the mobile client $C$ receives the message $aP \oplus H(PIN), EK_S$ from the gateway GW in step (2), it includes a fresh Diffie-Hellman element $aP$ by the server S. Therefore, the C must compute a fresh session key $sk$ by using the received $aP$ and a generated random number $b$. C then sends back an encrypted value $EK_S[bP, H(sk)]$ including another fresh Diffie-Hellman element $bP$ to S as a response. Note that $aP$ and $bP$ separately generated by C and S are fresh on each session and are different every time. Besides, $H(sk)$ in $EK_S[bP, H(sk)]$ and $bP$ in $E_{sk}[SN, bP]$ guarantee their integrity and source, respectively. In addition, it is impossible to create corresponding responses and their message authentication values, $EK_S[bP, H(sk)]$ and $E_{sk}[SN, bP]$, without knowing the shared secret password PIN between C and S. Since the C and S always verify the integrity of the fresh session key $sk$ by checking $H(sk)$ and $bP$, the replayed messages can be detected by the C and S, respectively. Therefore, except for $C$ and $S$, no one can pass the challenges. As a result, the proposed security mechanism can resist replay attacks.

5.2 **Resistance to Modification Attacks**

The proposed security mechanism resists modification attacks: An attacker Eve may modify the messages $aP \oplus H(PIN), EK_S, EK_S[bP, H(sk)]$, and $E_{sk}[SN, bP]$ being transmitted over an insecure network. However, although Eve forges them, the proposed security mechanism can detect this attack, because it can verify not only the equality of $sk$ computed by each party, but also the correctness of $EK_S[bP, H(sk)]$ and $E_{sk}[SN, bP]$ transmitted between two parties through validating $H(sk)$ and $bP$ in the security mechanism. Therefore, the proposed security mechanism resists modification attacks.

5.3 **Resistance to Password Guessing Attacks**

The proposed security mechanism resists password guessing attacks: An attacker Eve can intercept a message $aP \oplus H(PIN), EK_S$ sent by GW in step (2) over a public network. However, due to Definitions 3 and 5, he/she cannot derive the $C$’s secret password PIN from $aP \oplus H(PIN)$. Suppose that Eve intercepts $EK_S[bP, H(sk)]$ sent by C in step (3)
and $E_{sk}[SN, bP]$ sent by $S$ in step (4), respectively. Due to Definition 2, it is also extremely hard for $Eve$ to decrypt $EK_S[bP, H(sk)]$ and $E_{sk}[SN, bP]$ without knowing the $S$’s strong private key $DK_S$ and the session key $sk$. Therefore, the proposed security mechanism resists password guessing attacks.

5.4 Resistance to Man-in-Middle Attacks

The proposed security mechanism resists man-in-middle attacks: A mutual secret password PIN between $C$ and $S$ (or $GW$) is used to prevent the man-in-middle attack. The illegal attacker $Eve$ cannot pretend to be $C$ or $S$ (or $GW$) to authenticate the other since he/she does not own the mutual secret password $PIN$. Therefore, the proposed security mechanism resists man-in-middle attacks.

5.5 Resistance to Denning-Sacco Attacks

The proposed security mechanism resists Denning-Sacco attacks: In the proposed security mechanism, although an attacker $Eve$ obtains a fresh session key $sk = abP$, he/she cannot obtain the $C$’s secret password $PIN$ and the $S$’s private key $DK_S$ from the public channel values $aP \oplus H(PIN)$, $EK_S$ in step (2), $EK_S[bP, H(sk)]$ in step (3) and $E_{sk}[SN, bP]$ in step (4) because $DK_S$ is a strong secret key by Definition 1 and $H(PIN)$ is protected by $aP$. Although, $Eve$ can obtain $bP$ by decrypting $E_{sk}[SN, bP]$ with $sk$, he/she cannot get $aP$ from $bP$ and $abP$ due to the Definitions 3 and 4. Therefore, the proposed protocol can prevent the Denning-Sacco attack.

5.6 Evaluation of Secure Mutual Authentication

The proposed security mechanism provides secure mutual authentication: Mutual authentication means that both the client and server are authenticated to each other within the same security mechanism. Mutual authentication between $C$ and $S$ is achieved, because $C$ and $S$ authenticate each other with $H(sk)$ in the step (4) and $bP$ in the step (5), respectively. Nobody can create the $C$’s response value $EK_S[bP, H(sk)]$ without knowing the password $PIN$ and the shared common session key $sk$ between $C$ and $S$ and the $S$’s response value $E_{sk}[SN, bP]$ without knowing the private key $DK_S$ of $S$. In other words, it is infeasible for an attacker to masquerade as a legal client or a legal server. Also, the proposed security mechanism uses the Elliptic Curve Diffie-Hellman key exchange algorithm in order to provide mutual explicit key authentication. Then, the key is explicitly authenticated by a mutual confirmation session key $sk = abP$. Therefore, the proposed security mechanism provides secure mutual authentication.

5.7 Evaluation of Perfect Forward Secrecy

The proposed security mechanism provides perfect forward secrecy: In the proposed security mechanism, since the Elliptic Curve Diffie-Hellman key exchange algorithm is used to generate a session key $sk = abP$, perfect forward secrecy is ensured because an attacker with a compromised $C$’s secret password $PIN$ and $S$’s private key $DK_S$ are only able to obtain the $aP$ and $bP$ from an earlier session. In addition, it is also computationally infeasible to obtain the session key $abP$ from $aP$ and $bP$, as it is an ECDLP and an ECDHP. Therefore, the proposed security mechanism provides a perfect forward secrecy.

The security properties of related security mechanisms and the proposed security mechanism are summarized in Table 2.

6. Discussion about Computational Costs

This section discusses the computational costs of the Elliptic Curve Diffie-Hellman (ECDH) key exchange in the proposed security mechanism. In the proposed security mechanism, the server $S$ and the mobile client $C$ require two scalar multiplications of elliptic curve for ECDH key exchange, respectively. As we all know, the computational cost of ECDH is much larger than that of the secure hash function or XOR operation. Nevertheless, the ECDH operation requires to provide perfect forward secrecy in the proposed security mechanism. But the ECDH computations must not affect the use of the mobile device to which the resource was restricted.

In the proposed security mechanism, the ECDH computations do not matter to $S$ because $S$ has powerful computation abilities. However, the ECDH computations can influence to the mobile device of the client $C$. But we believe that the mobile device can compute the ECDH computations for secure mobile commerce transactions. As we all know, an ECC with 160-bit key length could offer roughly the same level of security as RSA with 1024-bit modulus. In the proposed security mechanism, one scalar multiplication of elliptic curve (i.e. $bP$) can be computed by $C$ in an off-line
manner. So the mobile device of $C$ only needs to perform one scalar multiplication of elliptic curve (i.e. $sk = abP$) in step (3). As a result, in view of efficiency computation, the proposed security mechanism is efficient to provide perfect forward secrecy since it does not involve costly digital signature, bilinear pairings and modular exponentiations.

Some previous implementations of elliptic curve cryptographic primitives on smart cards or microprocessors have been developed [12]–[16]. Recently, Scott et al. actually evaluate the cost of one scalar multiplication with the Philips HiPersmart card, where the processor of HiPersmart card offers a maximum clock of 36 MHz and 16 K RAM memory [15], [16]. In which, $G_1$ is a subgroup of order $q$ on an elliptic curve over a finite field $E(F_p)$, where $p$ is a 512-bit prime and $q$ is a 160 bit prime. Under this situation, the time spent in scalar multiplication of elliptic curve (i.e., the time spent in scalar multiplication of elliptic curve) is around 270 ms. It is obvious that the proposed security mechanism can not only solve the security flaws of the previous related security mechanisms for mobile commerce transactions and is applied to authenticate the mobile clients with limited computing capability.

In recent years, Wireless Application Protocol (WAP) has been gaining increasing popularity as a platform for mobile e-commerce; its security has thus become an important issue. In order to support the desired security feature perfect forward secrecy, and to resist various attacks, WAP offers the service of secure wireless data exchange of information thanks to a security protocol called Wireless Transport Layer Security (WTLS). WTLS supports the Elliptic Curve Cryptography (ECC) for secure session key establishment, where the client is typically a mobile device and the server a workstation that provides access to Internet in a wireless fashion. Therefore, we believe that the computational cost of ECDH in the proposed security mechanism does not affect serious problems for the use of the mobile device to which the resource was restricted.

7. Conclusion

This paper showed that both Yeh-Tsai’s and Yum et al.’s security mechanisms still do not provide perfect forward secrecy and are susceptible to a guessing attack and Denning-Sacco attack. In addition, we proposed a new security mechanism for mobile commerce transactions to overcome the weaknesses of the previous related security mechanisms. As a result, the proposed security mechanism not only remedies the weaknesses shown in both security mechanisms, but also greatly improves the robustness of security mechanism through secure mutual authentication and session key agreement.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2010-0010106).

References

Eun-Jun Yoon received his MSc degree in computer engineering from Kyungil University in 2002 and the PhD degree in computer science from Kyungpook National University in 2006, Republic of Korea. From 2007 to 2008, he was a full-time lecturer at Faculty of Computer Information, Daegu Polytechnic College, Republic of Korea. He is currently a 2nd BK21 contract professor at the School of Electrical Engineering and Computer Science, Kyungpook National University, Republic of Korea. His current research interests are cryptography, authentication technologies, smart card security, network security, mobile communications security, and steganography.

Kee-Young Yoo received his BSc degree in education of mathematics from Kyungpook National University in 1976 and the MSc degree in computer engineering from Korea Advanced Institute of Science and Technology in 1978, South Korea. He received the PhD degree in computer science from Rensselaer Polytechnic Institute in 1992, New York, USA. Currently, he is a professor at the Department of Computer Engineering, Kyungpook National University, South Korea. His current research interests are cryptography, authentication technologies, smart card security, network security, DRM security, and steganography. He has published over a hundred technical and scientific international journals on a variety of information security topics.